Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

TRANSFORMING DISCRETE EVENT MODELS TO MACHINE LEARNING MODELS

Hessam S. Sarjoughian
Forouzan Fallah
Seyyedamirhossein Saeidi

Arizona Center for Integrative Modeling and Simulation
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, 85281, USA

Edward J. Yellig

Intel Corporation
5000 W. Chandler Blvd
Chandler, AZ, 85226, USA

ABSTRACT

Discrete event simulation, formalized as deductive modeling, has been shown to be effective for studying dynamical systems. Development of models, however, is challenging when numerous interacting components are involved and should operate under different conditions. Machine Learning (ML) holds the promise to help reduce the effort needed to develop models. Toward this goal, a collection of ML algorithms, including Automatic Relevance Determination is used. Parallel Discrete Event System Specification (PDEVS) models are developed for Single-stage and Two-stage cascade factories. Each model is simulated under different demand profiles. The simulated data sets are partitioned into subsets, each for one or more model components. The ML algorithms are applied to the data sets for generating models. The throughputs predicted by the ML models closely match those in the PDEVS simulated data. This study contributes to modeling by demonstrating the potential benefits and complications of utilizing ML for discrete-event systems.

1 INTRODUCTION

Dynamical models are vital for understanding and operating complex systems, including smart manufacturing. Models must be able to simulate well-defined conditions, but the development efforts and computing resources can vary greatly based on system scale and complexity. A unifying concept for describing dynamical systems is to define them to have inputs, operations, and outputs (Wymore, 2018). Two well-known methods for model specifications are deductive and inductive. The former requires specifying structure and behavior abstractions for predicting output given some expected input. The latter extracts some system structure and behavior from expected input and output data.

A deductive model is one where its outputs are computed given some known operations that act on some inputs. These kinds of models are generally based on first principles. Operations are commonly defined in terms of states, functions, and relations that transform inputs into outputs. They are formalized according to some cause-and-effect formulations. Each model has its own structure and behavior. Considering the Parallel Discrete Event System Specification (PDEVS) modeling formalism (Chow and Zeigler, 1994), it is well-suited for describing and simulating the dynamics of discrete event systems as a collection of interacting components.

In inductive modeling, the operations for some observed actual and/or simulated input and output regimes are usually identified using second principles. Compared with first principles, these principles are not founded on axioms such as the laws of motion in Physics. Such operations are expected to be applied to unobserved input regimes to generate acceptable output regimes. Machine Learning Algorithms (MLA), such as Linear Regression and Naïve Bayes (Bishop and Nasrabadi, 2006), broadly fall into the inductive

modeling paradigm. The viability of MLA can vary significantly depending on the quantity and quality of a given system's input/output data and behavior under different operating conditions as well as algorithms' hyperparameters. Considering discrete event systems, their data and output regimes are time-dependent and may behave deterministically or not.

Development of component-based models for new systems generally requires significant time and resources including the skills to create knowledge (Fujimoto et al., 2018). Although componentization helps with restraining model complexity and understanding, making changes to models becomes difficult or impractical in view of expert skill, time, and effort. As the scale and complexity of models grow, more powerful computational platforms are needed. Development and use of such parallel and distributed simulation platforms can be exceedingly time-consuming and expensive. Even having such models in hand, it may be impractical to simulate them in a timely manner.

Machine Learning models can be difficult to generate as they depend on knowledge in specific domains. Large data sets and computing resources are generally needed for training and testing. Time in most ML models, particularly regression type, does not serve an explicit principal role. A substantial amount of time may be needed to develop these models. They are expected to predict output patterns given unobserved input patterns that closely mimic some observed combined input and output patterns. Still, the development of ML models is desirable since they hold the potential to be extracted from data instead of creating them from scratch from problem descriptions that may be unclear and incomplete. MLA may be executed much faster (inference time) and benefit from specialized computing platforms, including GPU clusters, when a vast amount of observed and/or simulated data is needed to extract hidden knowledge chiefly for analysis and, to a lesser degree, for prediction.

The above indicates deriving models using ML from component-based models stands to advance simulation studies under the assumption the necessary simulated data is available or otherwise can be obtained in a timely fashion. This research focuses on smart semiconductor manufacturing factories, a class of dynamical discrete event systems that have intricate structures and behaviors. Such factories with many kinds of machines rely on simulations to find efficient schedules for connected processes. Individual factories should enable the flexibility for producing computer chips through new kinds of connected factories. Simulations of these systems should be more flexible and efficient for models that are expected to grow in complexity and scale.

This paper details an approach where two discrete-event models of re-entrant semiconductor manufacturing factories are used and transformed to ML models. Sections 2 and 3 provide a short background on Parallel DEVS and supervised MLA and closely related works. Section 4 describes a method for transforming PDEVS models into ML regression models and demonstrates its use for re-entrant Single-stage and Two-stage cascade semiconductor factory models. Section 5 details experiments to evaluate the generated ML relative to the PDEVS models. Section 6 concludes the paper and highlights future research.

2 BACKGROUND

Discrete-event models are generally created using first principles as deductive methods whereas MLA as inductive methods rely on second principles. There exist several approaches, frameworks, and tools for specifying and simulating discrete event models. Similarly, many MLA have been developed. As models, they represent some form of correlation among some given input and their processing as output data. The PDEVS and MLA are chosen for this research. These are briefly described next.

2.1 Discrete Event Modeling

Considering component-based deductive modeling, the atomic PDEVS model specification has input, state, and output variables and time-based functions. The specification requires concise formulations for processing inputs, state transitions, and generating outputs. As primitive components, they can be hierarchically composed to create a variety of composite models. Well-formed input and output provide concise semantics for individual components and their compositions. As a deductive modeling approach, it

lends itself to concise understanding and causal reasoning. Input events may arrive and outputs may be produced with non-uniform time intervals. Multiple input and output events may occur concurrently at arbitrary time instances. State transition can occur either due to receiving input events (external transition function) or not (internal transition function). Furthermore, unlike classic DEVS (Discrete Event System Specification), concurrency achieved through a confluent function (i.e., some combination of the external and internal transition functions) is allowed. These models can be simulated using a variety of frameworks and tools developed in popular programming languages and executable on single and multi-processor computing platforms supported with distributed technologies (e.g., web services). The execution of these models conforms to the parallel simulation protocol where operations in the models with their input and output interactions are entirely observable. Model structures (components, couplings, and hierarchy), model types, atomic and coupled behavioral complexities, and primitive/compound data types result in the durations of simulation executions varying from a few milliseconds to hours and days. This becomes important when many simulation experiments are needed.

2.2 Machine Learning Algorithms

Numerous MLA are designed to extract models from available input and output data. The models make predictions that match unobserved data in varying degrees. As inductive models, such models define correlations between input and output variables. A simple linear model has a function with intercept and coefficient parameters. The function should satisfy some gradient descent measure such as the Root Mean Squared Error by finding the best values for the parameters for some given input and output data set. This model should minimize the difference between the true and predicted values for some outputs. This kind of model is holistic in contrast to the component-based modeling highlighted above. These algorithms are classified and evolved in past decades alongside the increasing amount of gathered data and huge advances in software and hardware. Some well-known regression MLAs selected for this research include Automatic Relevance Determination (ARD), Bayesian, Decision Tree (DT), K-Nearest Neighbors (KNN), LassoLars, Passive Aggressive Regression, Support Vector Regression (SVR), and TheilSen. The ML regression models are expected to exhibit better execution performance, require less development time, and be simpler to comprehend compared to neural networks, especially when there is limited data availability. Among these models, KNN, ARD, and DT showed superior performance for regression problems involving the prediction of continuous target variables. This research will primarily focus on the results obtained from these three algorithms. The following sections will discuss a few notable aspects of these models.

KNN operates by identifying the k closest neighbor points in the training set to the test point and predicting the target value by calculating the average based on these neighboring points. The selection of k is a significant hyperparameter that affects the balance between overfitting and underfitting. KNN has the advantage of being able to capture nonlinear relationships between features and targets, making it an excellent choice for small datasets. ARD estimates the relevance of each feature in predicting the target variable. While Bayesian linear regression regularizes all features identically, it is not well-suited when only a few features are significant. In contrast, the regularization performed by this algorithm is highly adaptive since all weights are regularized differently (Thirion et al., 2011). It assigns a prior distribution to the regression coefficients, which promotes sparsity, meaning many of them will be set to zero. Relevant features will have non-zero coefficients, while irrelevant features will have coefficients that are close to zero. DT uses a process involving a sequence of binary selections through traversing an input data tree structure. It is a non-parametric algorithm that operates by recursively partitioning the feature space into smaller regions based on the features' values. Each partition corresponds to a node in the tree, with the leaf nodes representing the predicted values for the corresponding region. One benefit is the ability to handle nonlinear relationships between features and targets. It can also handle missing values and categorical features. However, DT can overfit the training data, particularly if the tree is too deep or the minimum number of samples per leaf is too small (Bishop and Nasrabadi, 2006). The scikit-learn library (Pedregosa et al., 2011) is used for all the implementations of the MLA in this paper.

3 RELATED WORKS

The prominence of semiconductor manufacturing has attracted researchers to study the use of machine learning approaches in predictive simulation. Numerous studies have been conducted to address the challenges and opportunities presented by the data-driven digital landscape of semiconductor manufacturing (Kang et al., 2020). A new approach is proposed to predict and control cycle times in large-scale semiconductor manufacturing systems by considering the interaction between workcentres through global information such as release quantities, Work-In-Process (WIP), cycle time targets, and machine capacities (Barhebwa-Mushamuka et al., 2023). The approach involves local scheduling decisions guided by production targets, which are determined by a constraint-based optimization for cycle time targets. Another study proposes a system to predict the throughput time of production processes in manufacturing using AutoML (regression) (Hiller et al., 2022). Regression analysis and DTs are used to develop predictive models. The case study demonstrated the application possibilities of throughput time predictions based on the provided systematization, and highlighted differences in data availability and prediction quality. The authors suggest that feature engineering is an area that requires more research to further promote the practical use of predictive models. In another work, the factors influencing cycle time and patterns are identified to create ML predictive models for estimating cycle time (Meidan et al., 2011).

A combination of clustering and regression analysis to Identify patterns in historical data and build models that could forecast cycle time has also been developed (Backus et al., 2006). Regression trees are used for obtaining predictions for lots in production based on completed lots that are similar. They can handle both categorical and continuous variables without scaling. Deep learning to predict jobs remaining time during manufacturing execution based on production big data has also been studied (Fang et al., 2020). The proposed method uses a stacked sparse autoencoder to learn representative features from high-dimensional manufacturing big data to make accurate predictions. This study is applied in a large-scale job shop equipped with 44 machine tools producing 13 types of parts. ML techniques can also be used to identify factors affecting throughput in a semiconductor factory (Singgih, 2021). This study also contributed to the development of the Mini-Fab using Anylogic. It proposed a data collection scheme for the production control mechanism and classified the throughput into "good" and "bad" categories. Predicting throughput or cycle-time at high accuracy is not considered for purposes such as identifying input/output patterns.

Research works can also be viewed to focus on reducing the effort needed to develop simulation models and gaining simulation speed-up. Statistical regression modeling is proposed for the class of continuous systems that can be faithfully modeled according to the classic DEVS formalism (Saadawi et al., 2016). This work focuses on mapping the atomic models should be mapped to a holistic set of variables and functions. It is proposed execution of such predictive regression models to gain simulation speed-up. The aim is for the simulator to need fewer tasks to execute. In contrast, the work presented in this paper focuses on generating ML regression models that can execute much faster compared to PDEVS models.

Considering semiconductor wafer factories, it is shown queuing models can be created from discrete-event models (Seok et al., 2020). Simulation speed-up is achieved by replacing high-fidelity model components with low-fidelity queuing models while achieving a desirable degree of accuracy. This work uses queuing theory to derive coarse-grain abstractions from their fine-grain counterparts. This is in contrast to using ML methods instead of queuing theory. In the case of using queuing models, they are combined with high-fidelity (coarse-grain) models. The PDEVS models are transformed into ML models and are envisioned to be combined with time-based discrete-event models.

Earlier research shows the use of recurrent neural networks to create the structure and behavior for finite memory classic DEVS models from some finite observed input and output event data (Choi and Kim, 2002). Another work proposes using actual/simulated data to generate models with fixed input and output ports and combined with classic DEVS models (Antoine-Santoni et al., 2019). Additionally, reinforcement learning for Digital Twin is proposed for the classic DEVS formalism (David and Syriani, 2022). This work shows that simulation models can adapt automatically to the changes observed in systems.

The briefly described related works highlight the different concepts and the challenges of extracting models and gaining simulation speed-up. The basic idea of the approach detailed in the remainder of the paper is to determine what to consider and how to generate ML regression models given the knowledge of the atomic and coupled PDEVS components.

4 TRANSFORMING PDEVS MODELS TO ML MODELS

This research shows an approach for extracting models using MLA applied to the data obtained from sets of discrete event simulation experiments. The approach can be viewed as having five steps. First, PDEVS are developed and verified for correctness. Second, a set of experiments are devised, simulated, and validated. Third, the simulation data sets are processed and structured in terms of input (features) and output data (labels) sets to be used with MLA. Fourth, a collection of MLA is selected and employed to predict output variables for the input variables used for a Mini-Fab and Two-stage Cascade Mini-Fab. This step involves choosing training and test data sets. Fifth, the strengths of MLA are evaluated in terms of quantitative metrics and PDEVS and ML modelers' intuitions for the application domain of interest.

Two PDEVS models (Single-stage and Two-stage cascade semiconductor factories) are developed and simulated using the DEVS-Suite simulator (ACIMS, 2023). The input data sets are partitioned according to their scales. The training and test data are randomly and deterministically selected from the portioned data sets. The random choice of training and test data is for evaluating the accuracy of output predictions. The deterministic choices of training and test data are aimed at evaluating the MLA learning ability.

4.1 Simulation Models

A PDEVS Single-stage factory and a larger and more complex Two-stage cascade factory are developed. The Single-stage model is developed based on an existing benchmark (Kempf, 1994). A simplified component diagram for this model (named Mini-Fab) is shown in Figure 1(a). This factory consists of Diffusion, Implantation, and Lithography parts having machines A, …, E. Each part has a coordinator that dispatches wafers to one of the machines. The assignment of wafers to a machine is instantaneous (i.e., a simulation step consumes zero logical time).

The factory models can receive wafer lots called Product a (Pa), Product b (Pb), and Test wafer (Tw). The lots need to form a batch with a size of three prior to being chronologically processed through six steps across Diffusion, Implantation, and Lithography: steps 1 and 5 for machines A and B, steps 2 and 4 for machines C and D, and steps 3 and 6 for machine E (see Figure 1). The possible number of experiments defined in terms of (Pa, Pb, Tw) tuples is 218,700 of which 5,000 are considered suitable for this study since the rules for batching are that only one Tw lot can be included in the batch at most while Pa and Pb can be mixed in step 1, they cannot be mixed in step 5. Every machine is defined to operate in four consecutive phases: loading, processing, unloading, and transporting of wafer lots with an assigned period with some random uncertainty (e.g., 10%). Moreover, transducers are devised to collect data from the machines and coordinators in each part of the factory. These transducers have no influence on the components' operations and their couplings.

Table 1 lists the data that is collected from simulations. The timing and other properties of these models can vary. These models are used to simulate a set of experiments conducted for the Single-stage and Two-stage Cascade Mini-Fab factories. The Single-stage model has 15 atomic and 5 coupled models with 100 couplings. Four transducer models are used to measure and collect input, output, and state information at every simulation execution step.

A Two-stage cascade model is conceptualized and created based on the Mini-Fab model (see Figure 1(b)). The number of machines in Lithography is changed for coupling it with a replica of the Single-stage Mini-Fab factory. This model, named Stage-1 Mini-Fab, is connected to the replica of the Mini-Fab, named Stage-1 Mini-Fab. The resulting cascade model with 7 transducers (named Cascade Mini-Fab) has 28 atomic, 10 coupled components and 204 couplings. This cascade model will be used to generate simulation

scenarios that exhibit higher structure and behavior complexities. The number for the Pa and Pb can be {0,2,3,6,9,12,18,24,27,30,36,45,48,54,60,72,81,90} and for Tw can be {0,1,3,6,9,12,15,18,27}.

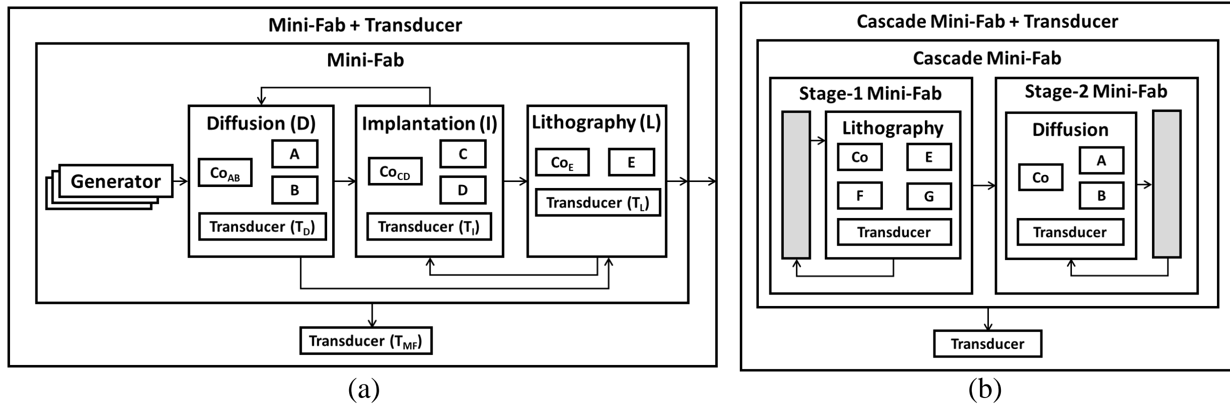


Figure 1: Component-based semiconductor factory models. (a) Single-stage Mini-Fab with feedforward and feedback connections. (b) Single-stage Mini-Fab factories form a Two-stage cascade factory.

Table 1: Machine, coordinator, and transducer components.

Diffusion	**Lithography**	**Mini-Fab**
• *A & B load times* • *A & B processing times steps* 1 & 5 • *A & B pransport times* • *A & B unload times* • *A & B processed lots in steps* 1 & 5 • *A & B throughputs* • Co_{AB} *steps* 1 & 5 *Throughputs* • *A & B turnaround times* • Co_{AB} *steps* 1 & 5 *turnaround times*	• *load time* • *processing times steps* 3 & 6 • *transport time* • *unload time* • *setup time* • *E processed lots in steps* 3 & 6 • *E throughput* • Co_E *steps* 3 & 6 *throughputs* • *E turnaround time* • Co_E *steps* 3 & 6 *turnaround times*	• *throughput time* • *turnaround time* Co_{AB}: Coordinator for Implantation component Co_E: Coordinator for Lithography component

4.2 Simulation Data Sets

For this research, 51 simulation experiments are devised and executed. The simulated data for the configured experiments are divided into three groups. The first has 24 configurations (called *Group-1*) with the total number of wafers being 6, 12, 18, 33, 36, to 54. The second has 16 configurations (called *Group-2*) with the total wafers being 54 and 72. The third has 11 configurations (called *Group-3*) with the total number of wafers being 90, 108, and 132. A total of 70 variables are measured via five machines, three coordinators, three generators, and four transducers for each of the 51 simulations. The logical time for executing these experiments ranges from 22,799 to 65,297 minutes. However, simulating and collecting all the data from some experiments can consume a significant amount of computing time when the total number of Pa, Pb, and Tw lots is 132 and the number of measured variables from Machines A, B, and E as well as their transducers for the Diffusion and Lithography components are large. The variables for Machines C

and D and their transducers are the same as for those for Machines A and B and their transducers. The collection of 51 simulations is considered a complete data set for use in MLA. Another set of experiments is conducted for the Two-stage cascade factory. A total of 142 variables are measured and collected.

The number of execution steps varies among simulation runs. For example, the experiment for (6,9,3) configuration completes after 397 steps. The experiments for (36,18,0) and (0-72-0) configurations are complete after 1,229 and 1,665 steps, respectively. This includes simulation steps for processing multiple input events and internal state transitions in zero-time logical simulation steps amongst all atomic models. The numbers of input and output events vary among the diffusion, implantation, and lithography components. The differences in the logic and timing of the operations in the machines can generate varied behaviors. The data sets are collected from the experiments where all received lots are fully processed for the Single-stage. In a few Two-stage experiments there are at most three unprocessed lots. The data collected at the end of each of the 51 simulation runs are used to generate and evaluate the ARD, DT, and KNN models.

5 EXPERIMENTS AND MODEL TRANSFORMATION EVALUATION

The use of the Mini-Fab PDEVS model for deriving ML models is evaluated by designing, developing, and analyzing the experiments highlighted in Section 4. The collected experimental data for a Single-stage Mini-Fab is divided into 18 scenarios corresponding to the modular structure shown in Figure 1. The scenarios for MLA have Diffusion D, Implantation I and Lithography L components with their corresponding transducers T_D, T_I, T_L and transducer T_{MF} for the Mini-Fab (see Table 2). These simulated data sets are named *Sim-Data-1*, …, *Sim-Data-18*. For example, nine MLA are applied to Sim-Data-10 with 28 features (i.e., the data collected from the Implantation, Diffusion, and Lithography components).

5.1 Single-stage Mini-Fab Factory

The predictions of a set of ML models trained using the common 80/20 split for training and testing, are compared with the actual simulation output. The 70/30, 60/40, and 50/50 ratios are also used for evaluating the nine MLAs (see Section 2.2). Table 3 shows the general Mean Absolute Error (MAE), Goodness of Fit (R^2), and Mean Square Error (MSE) measurements that are used to evaluate the Single-stage and Two-stage factory models. The results are similar to those obtained for the 80/20 training and test ratio. The results of the best three models (ARD, DT, KNN) as measured in terms of the factory throughput prediction errors are presented in Figure 2. The acronyms SD stand for the selection of the discrete-event machine and transducer components and FS for the number of features used in each scenario.

Table 2: Simulation experiments scenarios.

SD	Components	FS	SD	Components	FS	SD	Components	FS
1	D	14	7	$D+I$	13	13	$I+T_I+L+T_L$	35
2	$D+T_D$	22	8	$D+L$	36	14	$D+T_D+I+T_I+L+T_L$	57
3	I	14	9	$I+L$	22	15	$D+T_D+I+T_I+L+T_L+T_{MF}$	59
4	$I+T_I$	22	10	$D+I+L$	28	16	SD#15, excludes TH for D,I,L	51
5	L	44	11	$D+T_D+I+T_I$	35	17	SD#15, excludes TA for D,I,L	50
6	$L+T_L$	8	12	$D+T_D+L+T_L$	22	18	SD#16, excludes TA for D,I,L	39

Table 3: Measurements for the ARD, DT, and KNN Single-stage and Two-stage cascade models.

	Single-stage																		Two-stage			
SD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	1	2	3	4
MAE	4.04E-05	2.43E-05	6.37E-05	3.77E-05	2.43E-05	3.07E-05	3.99E-06	2.44E-05	2.04E-05	5.43E-05	1.69E-05	2.75E-05	1.64E-05	3.97E-06	3.97E-06	2.21E-05	3.99E-06	2.44E-05	8.88E-05	8.99E-05	5.97E-05	1.15E-05
R^2	0.995176	0.998898	0.991162	0.996039	0.998898	0.998449	0.999977	0.998895	0.999322	0.992157	0.999546	0.998505	0.999519	0.999978	0.999978	0.999241	0.999977	0.998895	0.986007	0.984204	0.992228	0.999801
MSE	5.E-09	1E-09	8.00E-09	3.73E-09	1.00E-09	1.46E-09	2.14E-11	1.00E-09	6.39E-10	7.00E-09	4.28E-10	1.41E-09	4.54E-10	2.12E-11	2.12E-11	7.15E-10	2.14E-11	1.00E-09	1.47-E08	1.66E-08	8.18E-09	2.08E-10
Mod	K = 1	K = 1	K = 1	D = 32	K = 1	D = 10	ARD	K = 1	D = 34	K = 1	D = 35	D = 14	D = 13	ARD	ARD	D = 37	ARD	K = 1	D =11 6	D = 18	D = 30	ARD

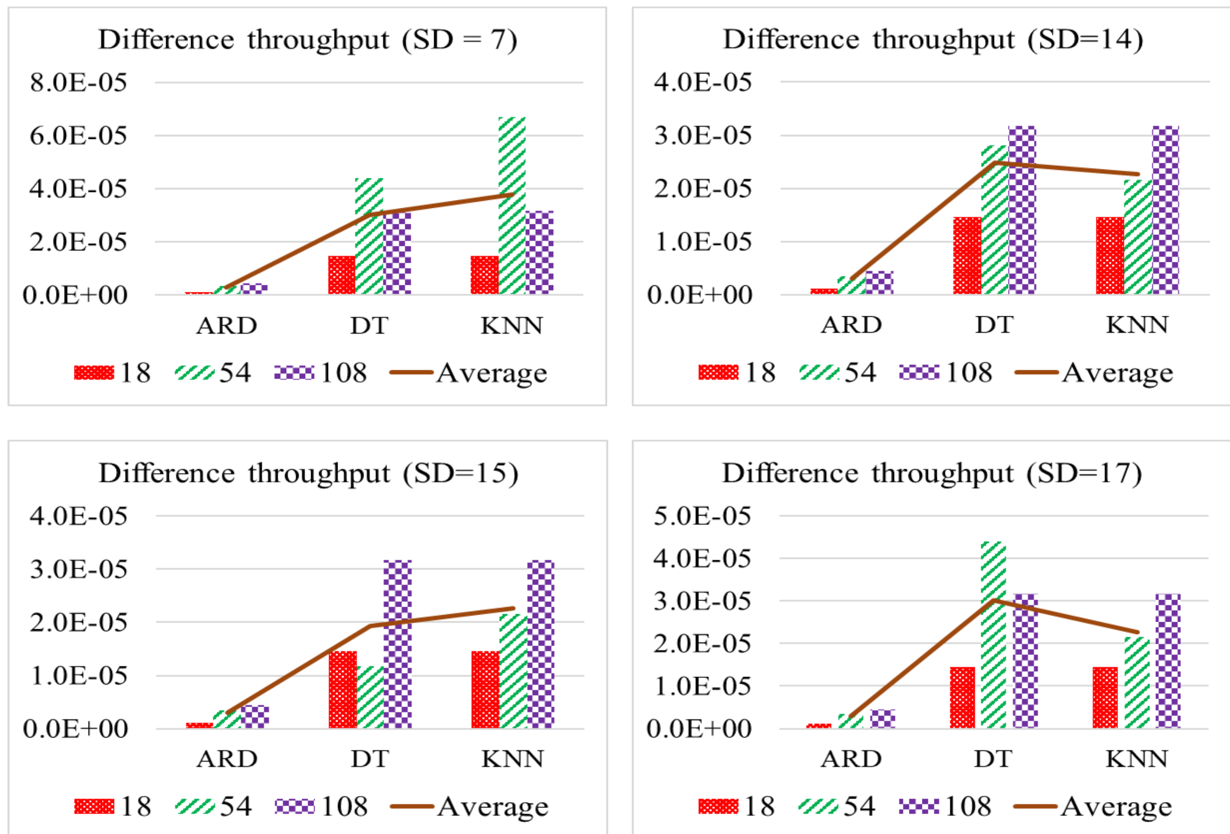


Figure 2: Comparison of the PDEVS model throughputs for the ARD, DT, and KNN models.

Table 4: Wafer configurations for evaluating the ARD model.

A1			B1			A2 & B2			C1			D1			C2 & D2		
Pa	Pb	Tw	Pa	Pb	Tw	Pa	Pb	Tw	Pa	Pb	Tw	Pa	Pb	Tw	Pa	Pb	Tw
9	9	0	27	18	9	12	18	6	18	27	9	54	0	18	30	45	15
12	6	0	27	27	0	54	0	0	18	36	0	54	30	15	45	45	0
18	0	0	24	36	12	36	18	0	54	0	0	90	90	27	60	60	12

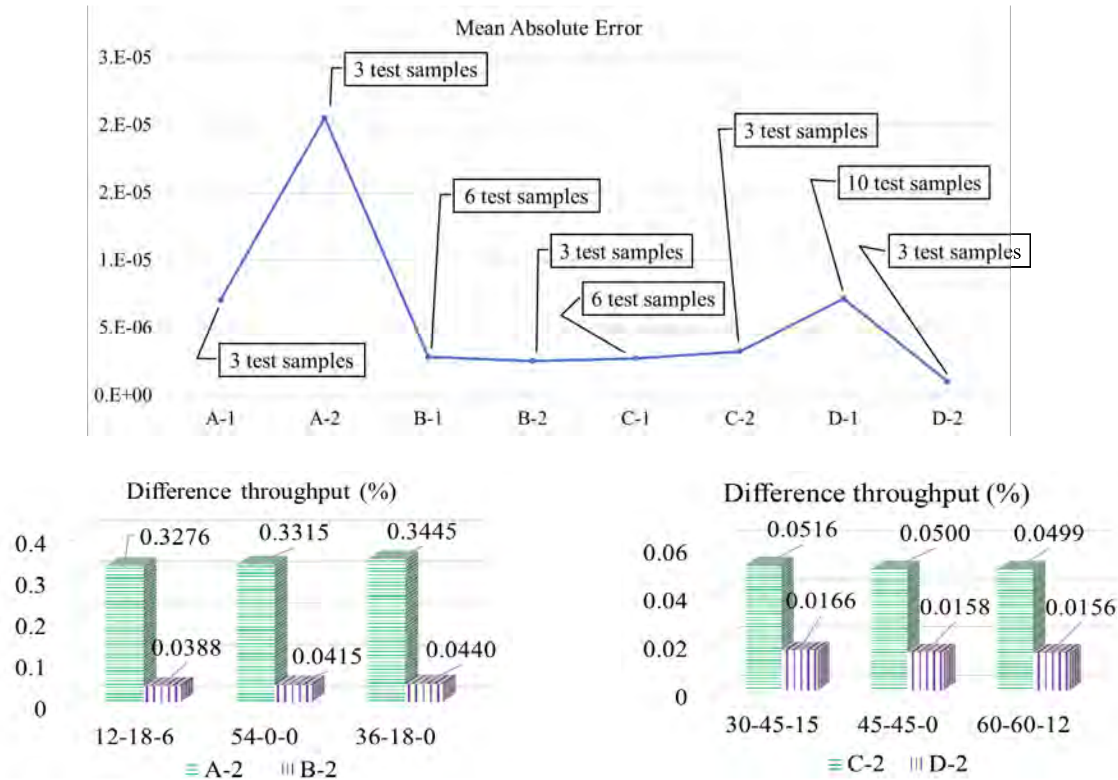


Figure 3: ARD model throughput predictions improve due to increasing the number of simulation scenarios.

Learning: Generating ML models involves using algorithms and statistical techniques to learn from data. One key aspect of this process is learning during the training phase. To understand and quantify the learning of ARD, a set of experiments is devised for selected demand profiles and training data sets. Four sets of experiment configurations are considered using 80% of data for the training set and 20% for the test set. The test demand profiles for the simulations are shown in Table 4. Two sets of experiments, namely A-1 and A-2 with a dataset consisting of 14 rows and B-1 and B-2 with a dataset of 27 rows, were conducted. For the testing phase of A-2 and B-2, three identical unseen test data points were employed. Figure 3 illustrates that the MLA exhibits improved performance as the training data increases. The MSE is reduced by 98.5%, while the R^2 value is increased by 11.2%. Following the same approach, other sets of experiments, namely C-1 and C-2 (which are devised similarly to B-1 and B-2, but with different unseen test data) and D-1 and D-2 are devised using 51 rows of the dataset. When comparing C-2 and D-2, the MSE is reduced by 89.9% and R^2 is increased by 0.7%. Since we have a limited number of simulations, the

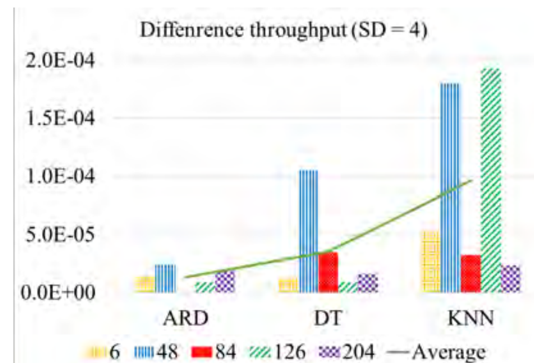
simulations don't contain the entire range of possible numbers of lots, and we simply skipped some in between. It should be noted that the range of number of lots in the test set is higher than the range in the training set, and this difference is significantly greater for D-1 and D-2 compared to C-1 and C-2, and even more so compared to A-1 and A-2 (see Section 4.2). Despite this difference in range, the ARD model shows it can still learn. Thus, given the exponential growth in the number of lots in simulation experiments, it is expected that the ARD model's learning will improve in accordance with the MAE, MSE, and R^2 measurements.

5.2 Two-stage Mini-Fab Factory

To further examine transforming PDEVS to ML models, the Single-stage simulation experimentation approach is carried out for the Two-stage Cascade factory model. The number of experiments for the Two-stage factory is 156 of which 125 are used for training and 31 for testing. The total numbers of Pa, Pb, and Tw for these simulations range from 6 to 204. Simulation executions process all wafers in 84 cases with the remaining 72 cases having 3 unprocessed wafers. For this study, four simulation data sets are selected using the model components from Stage-1 and Stage-2. $SD = 1$: Diffusion, Implantation, and Lithography with their respective transducers. $SD = 2$: same as (1) with turnaround time for the Two-stage Cascade Mini-Fab. $SD = 3$: Diffusion from Stage-1 and Lithography from Stage-2. $SD = 4$: All components in the Two-stage Cascade Mini-Fab except the lithography throughput and its throughput for steps 3 and 6 for Stage-2. The throughput prediction results for the ARD, DT, and KNN for wafer lots 6, 48, 84, 128, and 204 (see Figure 4(a)). Overall, the ARD performance (measured in throughput difference and MAE) is better than the DT and KNN models. Execution times are from five runs of the ML (ARD) and simulation models (PDEVS) on the same computer (see the table in Figure 4(b)). The ARD model is implemented in Python. The $\frac{Sim}{ML}$ measurements are the average execution times for different Pa, Pb, and Tw configurations. The execution performances of the ML models deserve further study.

$Lots$	$ARD\ (msec)$			$PDEVS\ (sec)$			$\frac{Sim}{ML}$
	Ave	min	max	Ave	min	max	
6	0.4	0.3	0.9	0.2	0.1	0.4	516
48	0.3	0.1	0.7	0.6	0.4	1.1	2,103
84	0.3	0.2	0.6	1.1	0.8	1.7	3,551
126	0.3	0.2	0.5	1.5	1.2	2.4	4,794
204	0.3	0.3	0.4	2.4	2.2	3.3	7,362

(a)



(b)

Figure 4: (a) Execution times for ARD and PDEVS models. (b) Throughput predictions for ARD, DT, and KNN for different Pa, Pb, and Tw configurations.

5.3 Discussion

The approach presented for transforming PDEVS models into ARD, DT, and KNN models is shown to be appropriate for Single-stage and Two-stage Mini-Fab factories. When transforming PDEVS models into these ML models, a relatively small amount of data is used. Additionally, the data sets have limited scope given the plethora of behaviors such factories may exhibit. Also, the factories have very small throughput, MAE, and MSE values with high R^2 values. In particular, the accuracy of the throughput predictions can

be difficult to understand and explain without having the simulation models. Therefore, it is important to consider the scale and range of values for the collected simulated data sets when generating and evaluating the viability of ML models. It should also be noted that the data from all the steps in each simulation were combined to represent a single data for use in the ML algorithms listed in Section x. By aggregating the information from all the steps in a simulation scenario, it is assumed the behavior of the model is continuous. Treating the dataset as continuous allows the use of continuous data modeling algorithms, including machine learning. These considerations suggest opportunities for further examination of the ML regression models and their viability for accurate predictions.

The machines and coordinators of the factories have non-deterministic behaviors. They have input/output events with arbitrary timing due to the randomness in the processing times assigned to the machines. Therefore, the time trajectory data sets collected from the simulations have irregular time durations. In other words, the simulation models are equipped to chronologically predict changes in the throughputs of individual machines and factories. The ARD, DT, and KNN algorithms, however, do not support time-based data sets. Therefore, these regression models do not predict throughput in time steps. In other words, it is not possible for these ML models to chronologically predict throughput as in PDEVS. Approaches such as Temporal Convolutional Networks (Lea et al., 2016) use time-series data sets. Time trajectory is treated as uniformly ordered unique numerical values (indexes), unlike discrete-event models.

6 CONCLUSION AND FUTURE RESEARCH

This research aims to utilize machine learning in studying and predicting the dynamics of discrete-event systems. This is achieved by developing two PDEVS factory models and transforming them into ARD, DT, and KNN regression models. These ML models are generated using limited sets of experiments obtained from the Single-stage and Two-stage Mini-Fab models with different wafer lot configurations. A key aspect of this approach is harnessing the inherent modularity of the PDEVS model specifications for generating the ML regression models. A simulation testbed is developed in the DEVS-Suite simulator for the Single-stage Mini-Fab factory model, followed by extending it to create a Two-stage Mini-Fab factory model. A scheme is developed to efficiently collect input, output, and state data from the factory models. The use of different ML regression models with various feature selections and knowledge of simulation models has produced throughput predictions that are accurate relative to their PDEVS simulation counterparts. This study shows some ML regression models can be used to predict the key throughput characteristic of the Min-Fab factory models. The ML models may offer a basis for future connected factories that require accurate predictions supported while achieving many-order speed-up execution time. Future research includes developing connected factory models under different structural topologies, operating conditions, and interaction modalities. Other kinds of ML models, including neural networks, should be explored to handle variations in the behaviors of models and enable chronological predictions of time-sensitive input, output, and state of the factories. It is also important to develop the means for composing PDEVS and ML models while achieving accelerated execution.

ACKNOWLEDGMENTS

This research is funded by Intel Corporation, Chandler, Arizona, USA. We are thankful to the reviewers who provided helpful critiques and suggestions on an earlier version of this paper.

REFERENCES

ACIMS. 2023. *DEVS-Suite Simulator, Version 7.0*. Arizona Center for Integrative Modeling and Simulation. https://acims.asu.edu/devs-suite/, accessed 21st March.

Antoine-Santoni, T., B. Poggi, E. Vittori, H. Van Hieux, M. Delhom, and A. Aiello. 2019. "“Smart Entity”– How to Build DEVS Models from Large Amount of Data and Small Amount of Knowledge?". *Simulation Tools and Techniques: 11th International Conference*. 615–626.

Backus, P., M. Janakiram, S. Mowzoon, G. C. Runger, and A. Bhargava. 2006. Factory Cycle-time Prediction with a Data-mining Approach. *IEEE Transactions on Semiconductor Manufacturing*. *19*(2): 252–258.

Barhebwa-Mushamuka, F., S. Dauzère-Pérès, and C. Yugma. 2023. "A Global Scheduling Approach for Cycle Time Control in Complex Manufacturing Systems". *International Journal of Production Research*. *61*(2): 559–579.

Bishop, C. M. and N. M. Nasrabadi. 2006. *Pattern Recognition and Machine Learning*. New York: Springer.

Choi, S. J., and T. G. Kim. 2002. "Identification of Discrete Event Systems Using the Compound Recurrent Neural Network: Extracting DEVS from Trained Network". *Simulation: Transactions of The Society for Modeling and Simulation International*. *78*(2), 90-104.

Chow, A. C. H., and B. P. Zeigler. 1994. "Parallel DEVS: A Parallel, Hierarchical, Modular Modeling Formalism". In *Proceedings of 1994 Winter Simulation Conference,* Edite by D. A. Sadowski, A. F. Seila, J. D. Tew, and S. Manivannan. 716–722. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc..

David, I., and E. Syriani. 2022. "DEVS Model Construction as a Reinforcement Learning Problem". In *Proceedings of 2022 Annual Modeling and Simulation Conference (ANNSIM)*. 30–41.

Fang, W., Y. Guo, W. Liao, K. Ramani, and S. Huang. 2020. "Big Data Driven Jobs Remaining Time Prediction in Discrete Manufacturing System: A Deep Learning-based Approach". *International Journal of Production Research*. *58*(9). 2751–2766.

Fujimoto, R., J. Barjis, E. Blasch, W. Cai, D. Jin, S. Lee, and Y.-J. Son. 2018. "Dynamic Data Driven Application Systems: Research Challenges and Opportunities". In *Proceedings of 2018 Winter Simulation Conference,* edited by B. Johansson, S. Jainm, O. Rose, M. Rabe, Skoogh A., N. Mustafee, and A. A. Juan, 664–678. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc..

Hiller, T., L. Deipenwisch, and P. Nyhuis. 2022. "Systemising Data-driven Methods for Predicting Throughput Time within Production Planning & Control". *IEEE International Conference on Industrial Engineering and Engineering Management*. 716–721.

Kang, Z., C. Catal, and B. Tekinerdogan. 2020. "Machine Learning Applications in Production Lines: A Systematic Literature Review". *Computers & Industrial Engineering*. 149: 106773.

Kempf, K. G. 1994. "Detailed Description of a Two-product Five-machine Six Step Re-entrant Semiconductor Manufacturing System". Intel Corporation, Technology & Manufacturing Group.

Meidan, Y., B. Lerner, G. Rabinowitz, and M. Hassoun. 2011. "Cycle-time Key Factor Identification and Prediction in Semiconductor Manufacturing using Machine Learning and Data Mining". *IEEE Transactions on Semiconductor Manufacturing*, *24*(2): 237–248.

Saadawi, H., G. A. Wainer, G. Pliego, S. Paul. 2016. "DEVS Execution Acceleration with Machine Learning". In *Symposium on Theory of Modeling and Simulation (TMS-DEVS)*. 1–6. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc..

Seok, M. G. I., C. W. Chan, W. Cai, H. S. Sarjoughian, and D. Park. 2020. "Runtime Abstraction-Level Conversion of Discrete-Event Wafer-fabrication Models for Simulation Acceleration". In *Proceedings of 2020 Principles of Advanced Discrete Simulation (PADS)*, 83–92.

Singgih, I. 2021. "Production Flow Analysis in a Semiconductor Fab using Machine Learning Techniques". *Processes*, *9*(3), 1–18.

Thirion, B., V. Michel, E. Eger, and C. Keribin. 2011. "Multiclass Sparse Bayesian Regression for fMRI-based Prediction". *International Journal of Biomedical Imaging*. 1–13.

Wymore, A. W. 2018. *Model-based Systems Engineering.* CRC Press.

AUTHOR BIOGRAPHIES

HESSAM S. SARJOUGHIAN is an Associate Professor of Computer Science and Computer Engineering in the School of Computing and Augmented Intelligence (SCAI) at Arizona State University (ASU), Tempe, Arizona. His research interests include model theory, poly-formalism modeling, collaborative modeling, simulation for complexity science, and M&S frameworks/tools. He is the co-director of the Arizona Center for Integrative Modeling and Simulation (https://acims.asu.edu). He can be reached at hessam.sarjoughian@asu.edu.

FOROUZAN FALLAH is a Ph.D. student in the Computer Science program in the School of Computing and Augmented Intelligence (SCAI) at Arizona State University, Tempe, AZ, USA. She can be reached at ffallah@asu.edu.

SEYYEDAMIRHOSSEIN SAEIDI is a Ph.D. student in the Computer Science program in the School of Computing and Augmented Intelligence (SCAI) at Arizona State University, Tempe, AZ, USA. He can be reached at ssaeidi@asu.edu.

EDWARD J. YELLIG is the director of Operational Decisions Support Technology at Intel Corporation. He has been with Intel for 26 years and has a Ph.D. in Operations Research with the emphasis in discrete event modeling of large-scale systems. His focus has been in developing fab models for determining capital requirements and is also responsible for the real-time digital twin tactical models. He can be reached at edward.j.yellig@intel.com.